МТС ods.ai

Your Second RecSys

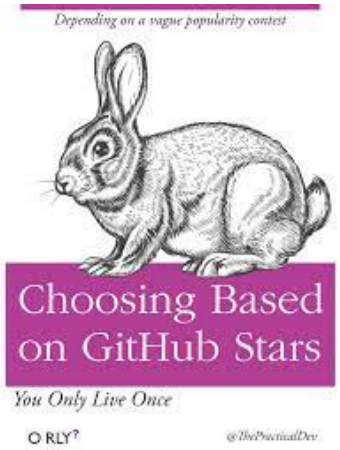
Ускорение рекомендаций в проде

Александр Бутенко

ML-разработчик
MTC BigData

План

- Вспомним про существование прода
- Приближенный поиск соседей – зачем он нам
- Готовые реализации приближенного поиска
- Кэширование – тоже ускорение

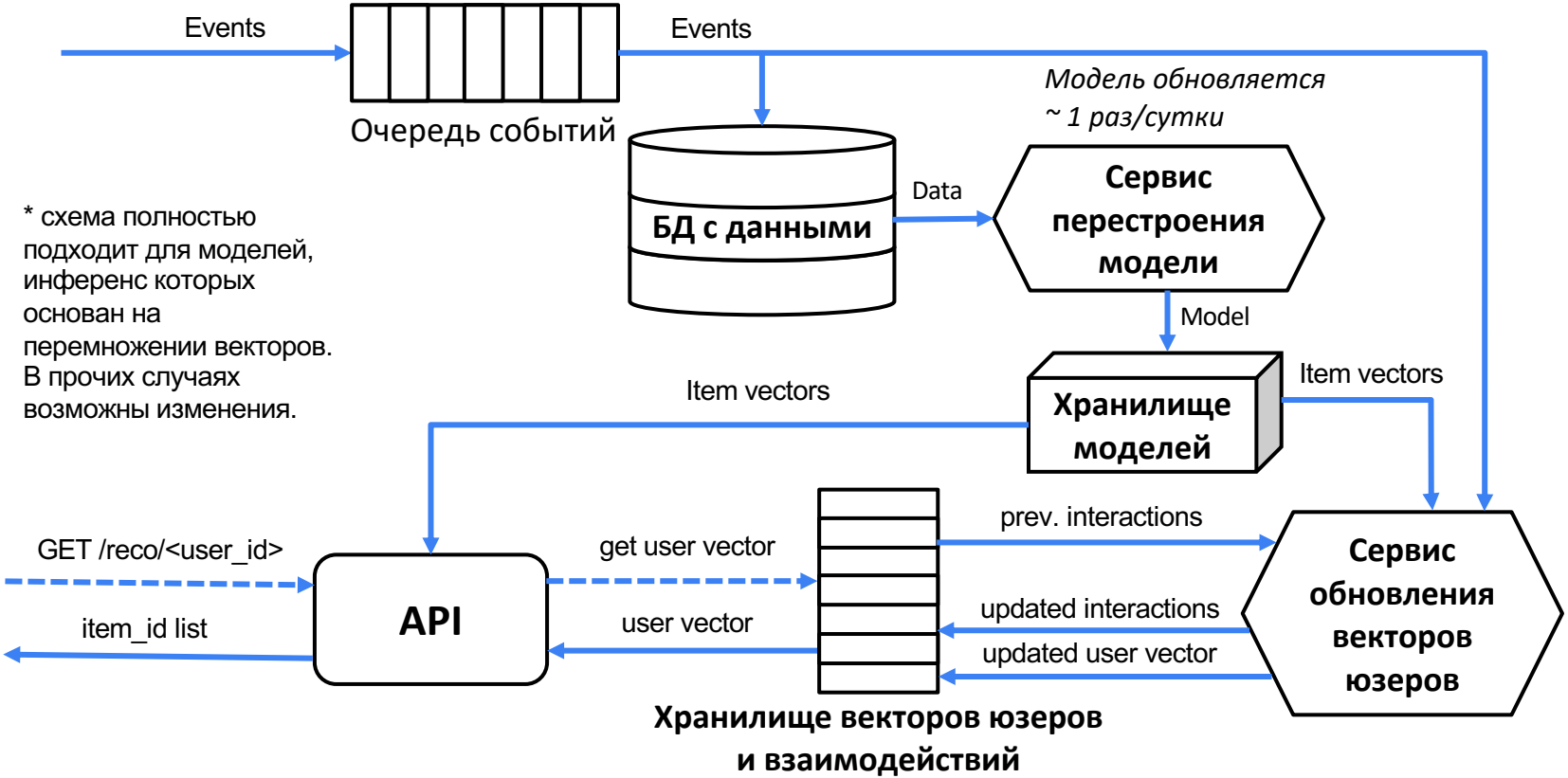


В предыдущих сериях

Вы узнали, что в проде рекомендации можно делать, используя подходы разной сложности:

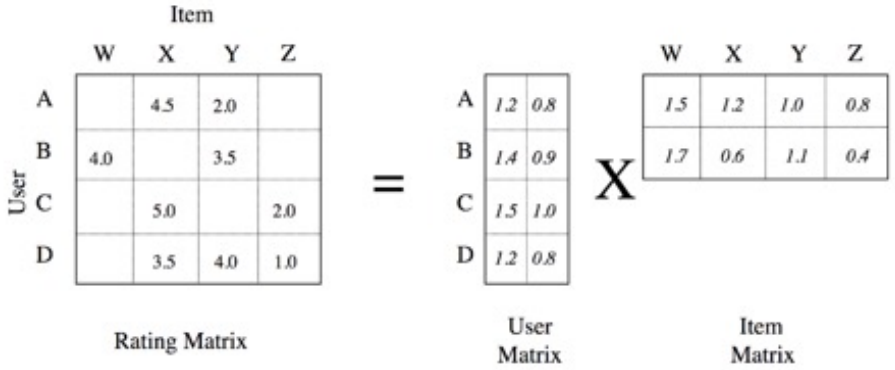
- Offline
- Nearline
- Online

Вспомним вариант Nearline



Самый частый кейс: векторные модели

Почему?



$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{x}_u \cdot \mathbf{y}_i \ ,$$

Самый частый кейс: векторные модели

Однако

- В реальном мире рекомендательная система должна соответствовать требованиям по времени ответа
- Чем больше у нас товаров/услуг и размерность векторов, тем медленнее вычисление и сортировка

Что делать?

- Approximate nearest neighbors, или приближенный поиск соседей
- Придумать что-то свое и написать статью

Nearest neighbor search

Задача поиска ближайших соседей

- Оптимизационная проблема поиска точки из некоторого множества точек, ближайшей к заданной точке
- Близость обычно измеряют некоторой функцией похожести
- Простейшая реализация – линейный поиск среди всех точек множества

Approximate nearest neighbor search (далее ANN) методы позволяют

- Свести линейный поиск среди всех точек к обходу определенной структуры данных
- Структура позволяет находить соседей быстро, но за это придется заплатить точностью

Переход от поиска max inner product к поиску соседей в евклидовом пространстве

В общем случае матричной факторизации прогноз считается как **inner product**:

$$r_{ui} = \mu + b_u + b_i + x_u \cdot y_i$$

Однако, реализации ANN в своей массе позволяют искать соседей, используя **L2** норму.

Ребята из Microsoft показали, что существует такая **трансформация векторов**, которая

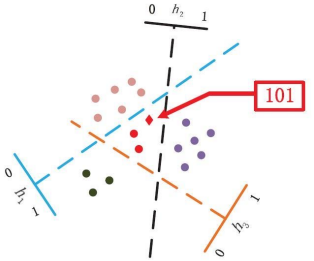
- Позволяет свести проблему поиска по inner product к проблеме поиска соседей в евклидовом пространстве с помощью L2 нормы
- Сохраняет порядок следования ближайших соседей

Таким образом, это преобразование позволяет нам для произвольной матричной факторизации воспользоваться эффективными методами поиска соседей

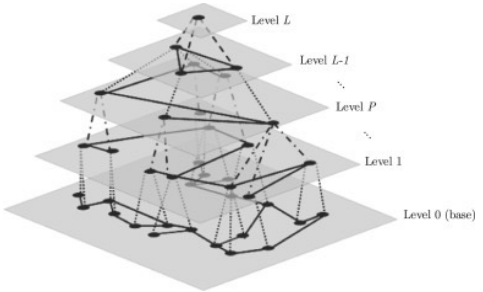
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/XboxInnerProduct.pdf

Некоторые популярные подходы

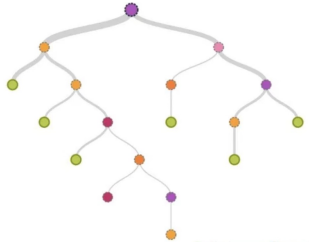
- С использованием семейств хэш-функций



- С использованием графов или иерархий графов



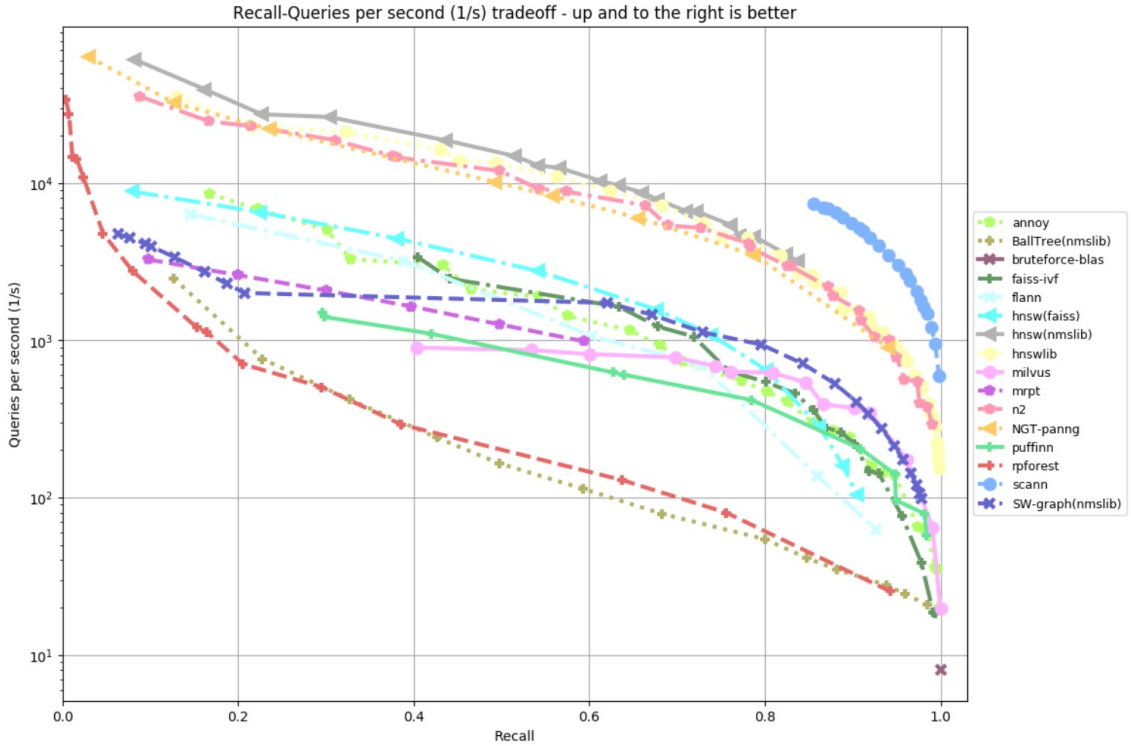
- С использованием деревьев или ансамблей деревьев



Популярные реализации

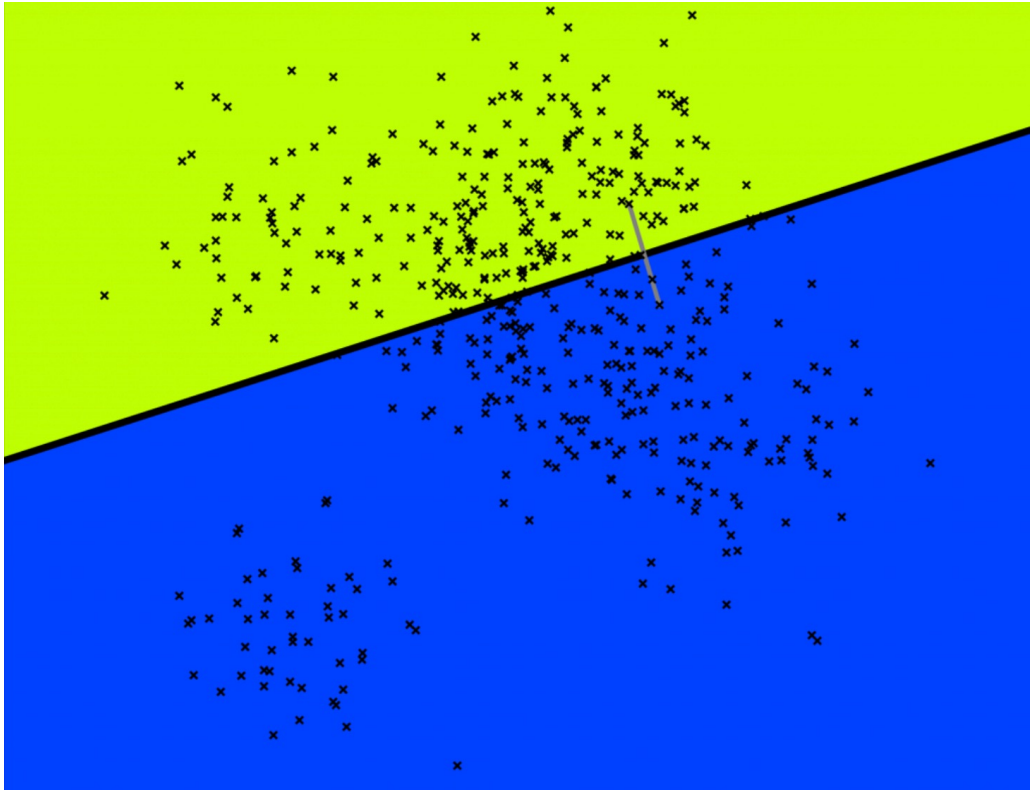
- Nmslib (cpu only)
- Annoy (cpu only)
- Faiss (cpu + gpu)
- Scann (cpu only)

Выбор лучше делать экспериментально.
Практика показывает, что перформанс зависит от датасета.
Референс:
http://ann-benchmarks.com/

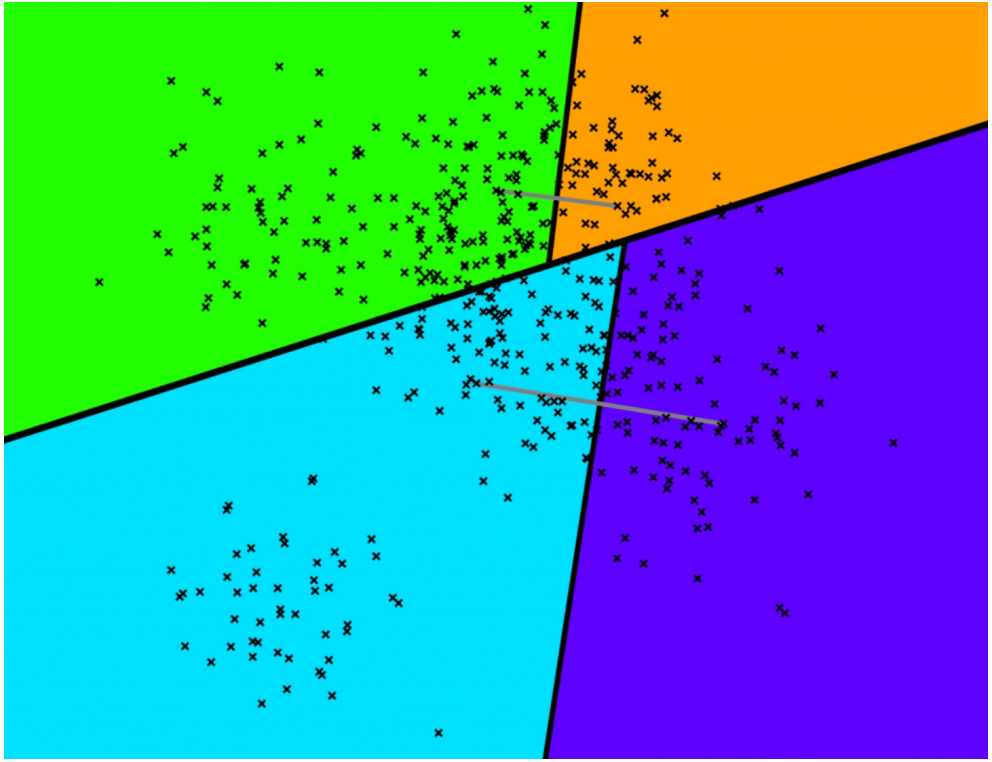


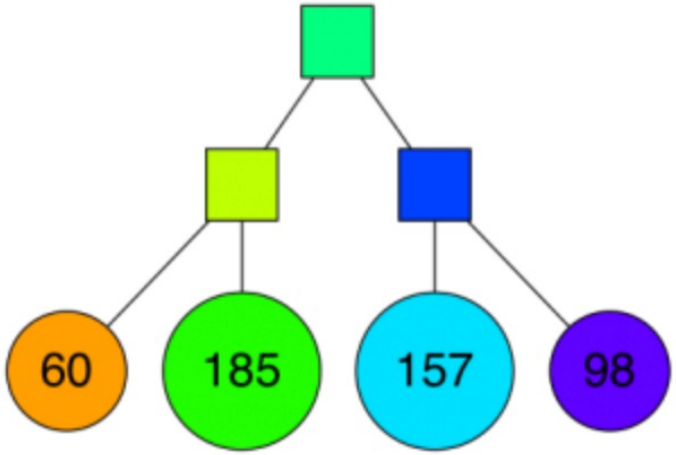
Пример: Annoy

Пример: Annoy

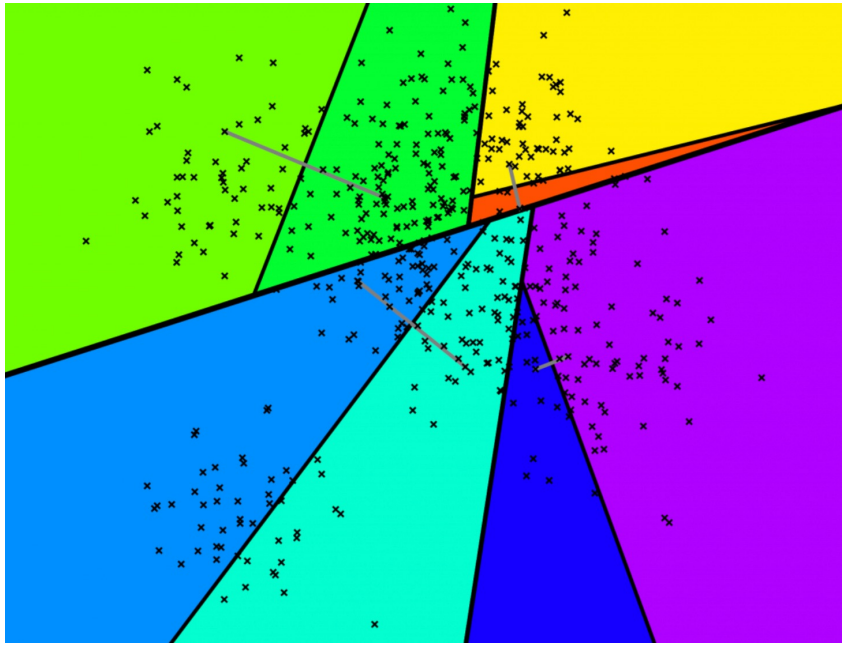


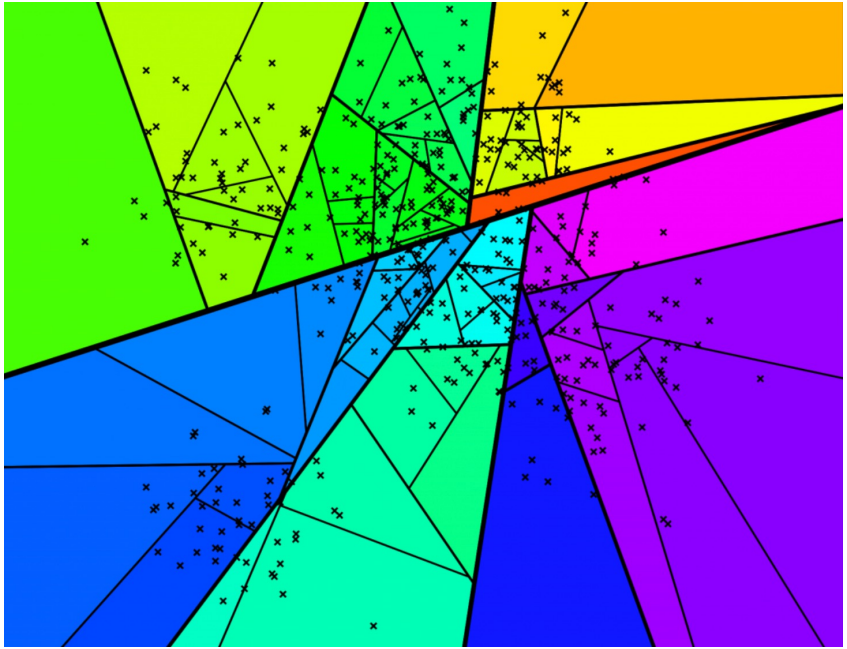
Пример: Annoy



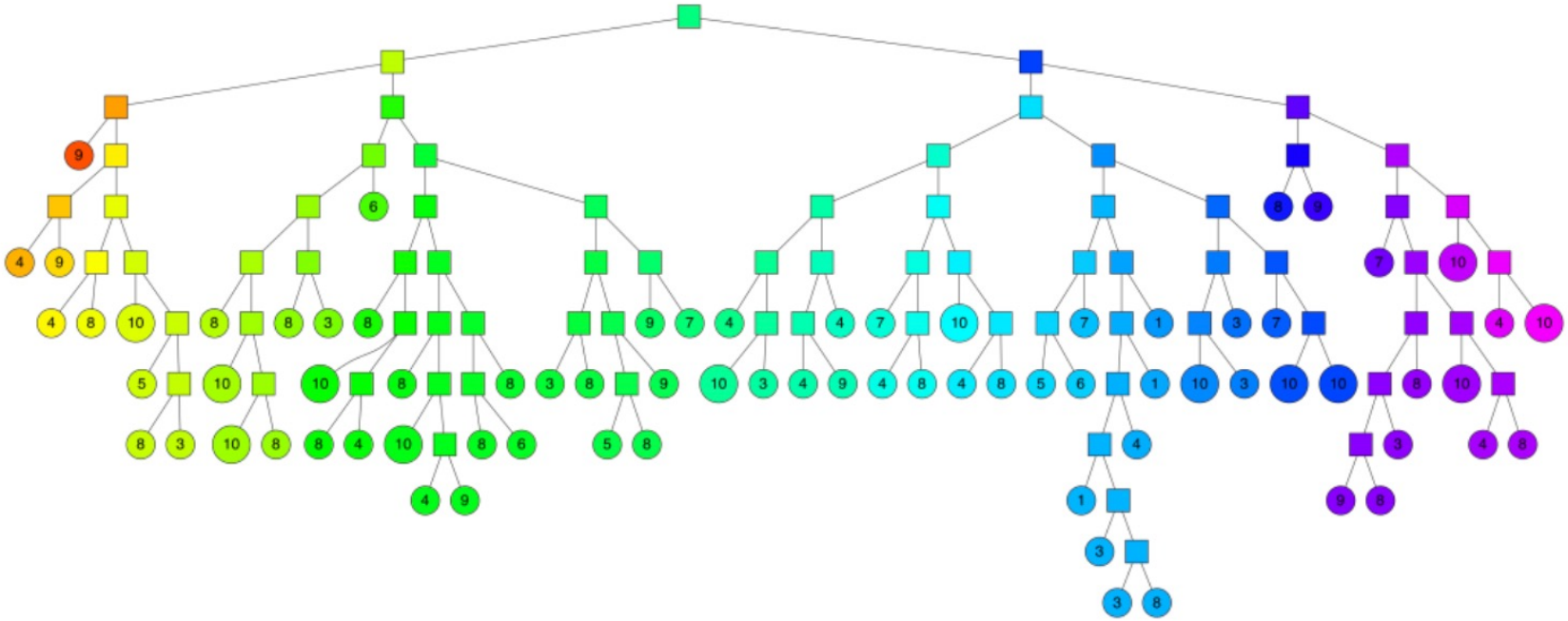


Пример: Annoy

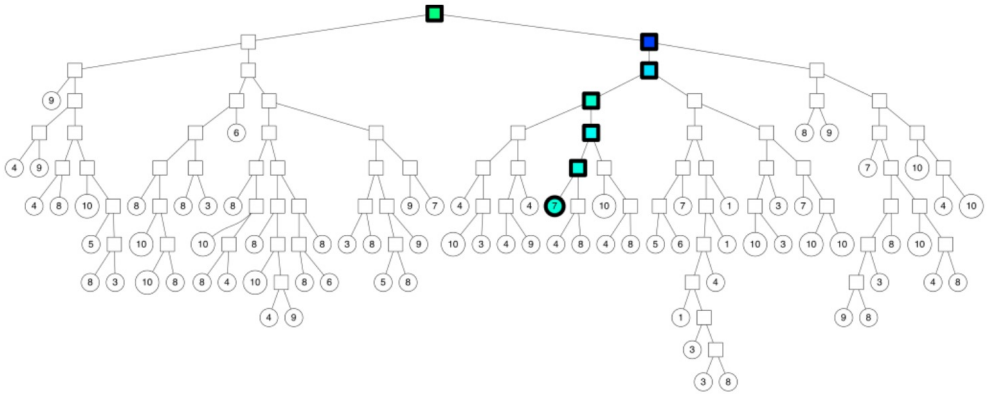


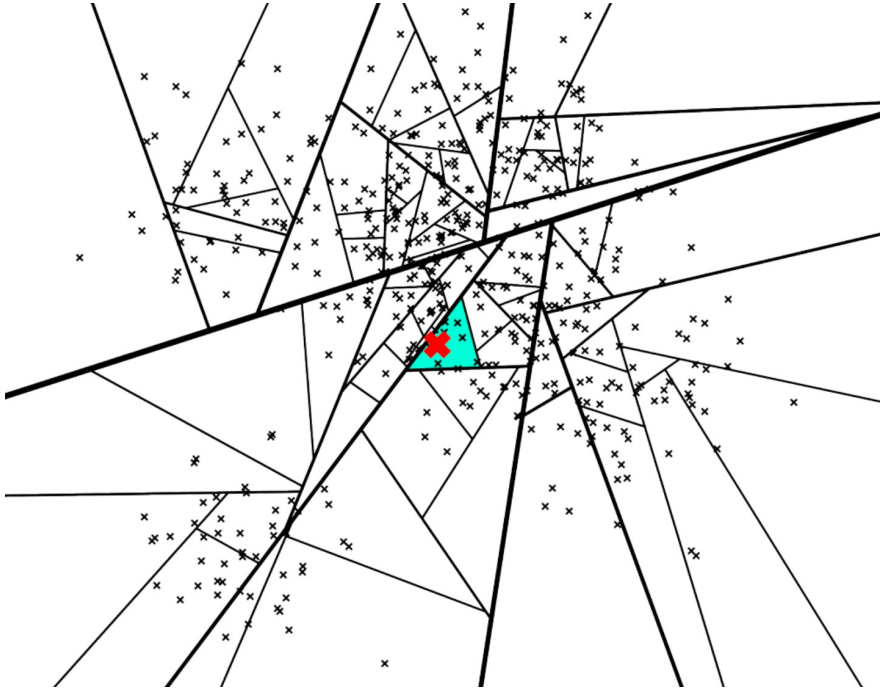


Пример: Annoy

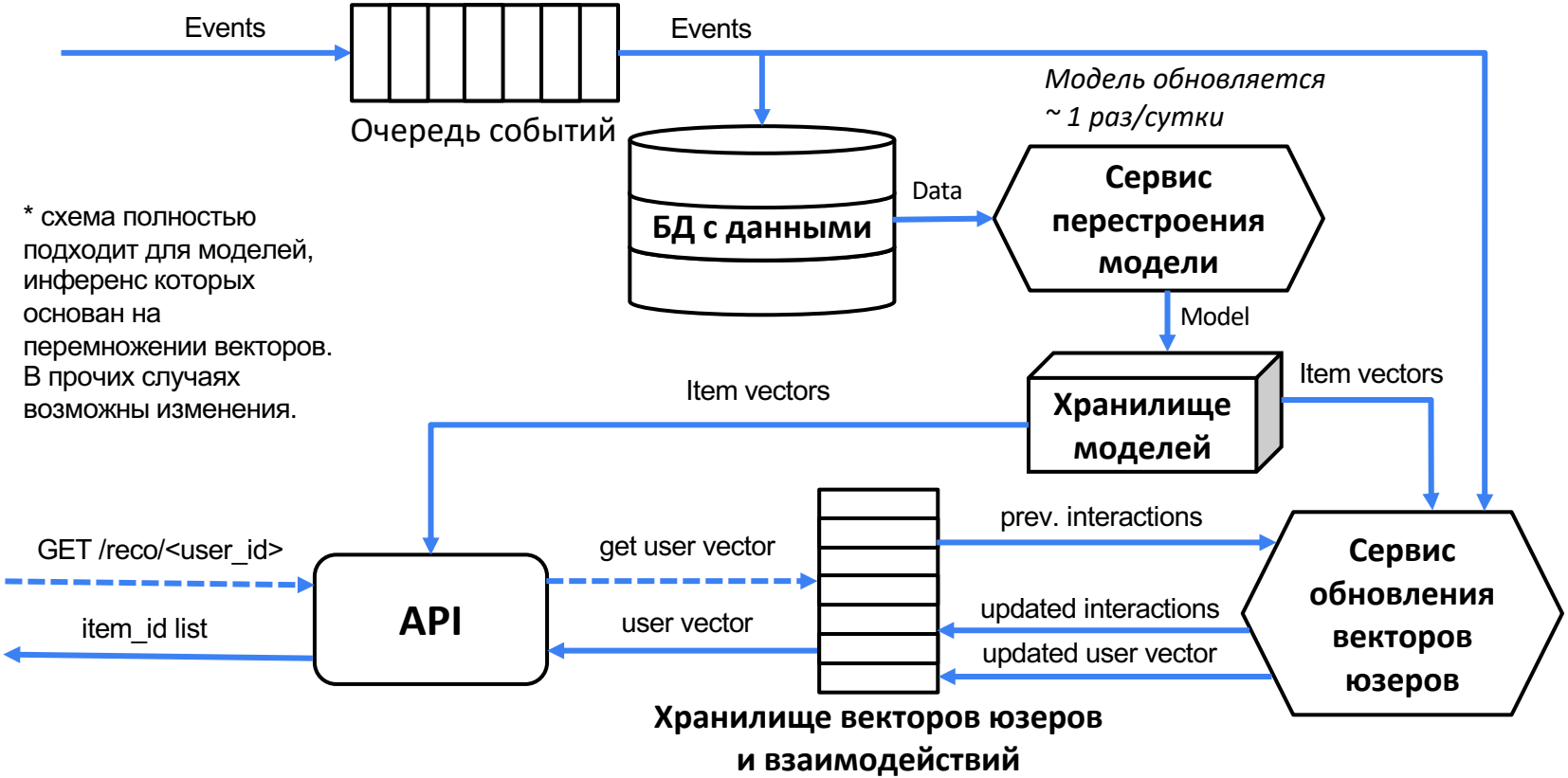


Пример: Annoy





Снова вспомним вариант Nearline



Trade off в реальных системах

- Нужно больше человеческих ресурсов на разработку
  - Сложный ci/cd
  - Множество разносортных хранилищ
  - Мониторинг и логирование
- Нужно продумать offline fallback на случай поломки
- Т.к. вычисления происходят на лету, сложно навесить многослойную логику постпроцессинга
- Больше источников ошибок в процессе доставки новой модели
- Скорее всего придется выбирать между большей скоростью и меньшей точностью
- Сложно реализовать фильтрацию

Мы это сделали, но все равно медленно

Кэширование – еще один способ ускорения

Что можно кэшировать?

Общий ответ - сильно зависит от конкретной задачи и узких мест в перформансе.

Например:

- Предрасчитанные рекомендации между обновлениями модели
- Предрасчитанные фильтры между обновлениями фильтров
- Если формируем целые страницы с рекомендациями, можно кэшировать части страницы, которые не требуют обновления на каждый запрос

Как кэшировать?

- Redis, Cassandra, Memcached
- Hash table

Кратко о важном

- Не стесняйтесь экспериментировать с разными реализациями ANN и выбирайте наиболее подходящую вам
- Не стесняйтесь кэшировать медленные вычисления. Соблюдайте баланс между сложно и быстро
- Не забывайте о том, что ANN дает приближенное решение
- Не оптимизируйте, если в этом нет нужды

Контакты

Александр Бутенко

a.butenko.o@gmail.com

@iomallach

https://www.linkedin.com/in/alexander-butenko-657b62187/